解析Java类和对象的初始化过程Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022__E8_A7_A3_E6_9E_90Java_c104_644639.htm 类的初始化和对象初始化是 JVM 管理的类型生命周期中非常重要的两个环节，Google 了一遍网络，有关类装载机制的文章倒是不少，然而类初始化和对象初始化的文章并不多，特别是从字节码和 JVM 层次来分析的文章更是鲜有所见。 本文主要对类和对象初始化全过程进行分析，通过一个实际问题引入，将源代码转换成 JVM 字节码后，对 JVM 执行过程的关键点进行全面解析，并在文中穿插入了相关 JVM 规范和 JVM 的部分内部理论知识，以理论与实际结合的方式介绍对象初始化和类初始化之间的协作以及可能存在的冲突问题。 问题引入 近日我在调试一个枚举类型的解析器程序，该解析器是将数据库内一万多条枚举代码装载到缓存中，为了实现快速定位枚举代码和具体枚举类别的所有枚举元素，该类在装载枚举代码的同时对其采取两种策略建立内存索引。由于该类是一个公共服务类，在程序各个层面都会使用到它，因此我将它实现为一个单例类。这个类在我调整类实例化语句位置之前运行正常，但当我把该类实例化语句调整到静态初始化语句之前时，我的程序不再为我工作了。 下面是经过我简化后的示例代码： ［清单一］ package com.ccb.framework.enums.import java.util.Collections.import java.util.HashMap.import java.util.Map.public class CachingEnumResolver { //单态实例 一切问题皆由此行引起 private static final CachingEnumResolver SINGLE_ENUM_RESOLVER = new CachingEnumResolver().